IVONNE: AN INTERACTIVE NETWORK MODEL-BUILDING SYSTEM.

Charles Stephen Burchinal

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IVONNE: AN INTERACTIVE NETWORK MODEL-BUILDING SYSTEM

by

Charles Stephen Burchinal

September 1981

Thesis Advisor: G. G. Brown

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
IVONNE: An Interactive Network Model-Building System

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; September 1981

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Charles Stephen Burchinal

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
September 1981

13. NUMBER OF PAGES
58

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Networks
Model-Building
Interactive Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Fast and efficient mathematical programming routines have been developed for network flow problems, but due to their complexity the average manager or lay analyst does not possess the mathematical programming backgroung required to construct the models or use the solution technology available. This problem is solved here by the development of an interactive network generating system designed to create, update, and solve

#20 - ABSTRACT - CONTINUED

a single-commodity network with only a minimal knowledge of network structure and only a rudimentary mastery of computer terminal use. This is accomplished through the interactive use of a set of FORTRAN programs which lead the user, step-by-step, through the construction of the network by a series of queries, and which links with GNET, a machine independent FORTRAN program for the solution of capacitated network flow problems.

Approved for public release, distribution unlimited.

IVONNE: An Interactive Network Model-Building System

by

Charles Stephen Burchinal
Captain, United States Marine Corps
B.A., Otterbein College, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1981

ABSTRACT

Fast and efficient mathematical programming routines have been developed for network flow problems, but due to their complexity the average manager or lay analyst does not possess the mathematical or programming background required to construct the models or use the solution technology available. This problem is solved here by the development of an interactive network generating system designed to create, update, and solve a single-commodity network with only a minimal knowledge of network structure and only a rudimentary mastery of computer terminal use. This is accomplished through the interactive use of a set of FORTRAN programs which lead the user, step-by-step, through the construction of the network by a series of queries, and which links with GNET, a machine independent FORTRAN program for the solution of capacitated network flow problems.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

The system name IVONNE has been chosen in appreciation of the author's wife, Ivonne Burchinal, who provided encouragement and secretarial skills necessary for successful completion of this thesis.

I. INTRODUCTION

Over the past two decades, research in optimization of network flow models has resulted in the development of fast and efficient mathematical programming routines for the solution of this class of problems. Coupling this with the ready availability of high speed computers, network problems long viewed as impracticable are now routinely used by management.

Due to the complexity of these algorithms and the intricacy of the models required to use them, the average manager or lay analyst frequently does not possess the mathematical or programming background required to construct the models or use the solution technology available. Thus, technically trained support personnel are required to construct the models, implement the appropriate routines, and make modifications. As a result, the decision maker or individual for whom the system was designed is insulated from it. This provides the major impetus for this thesis --- to create an interactive general network model-building and optimization system that can be used by an individual who has only a basic knowledge of network structure and a general understanding of computer terminal use.

Currently, industry is using network model-building and optimization systems as managemant support systems. For example, Hunt-Wesson Foods uses a multicommodity distribution planning system based on a multicommodity network model [Ref. 1] and has demonstrated effective application of the system [Ref. 2]. But as is common with all the models of this type, most of the interface facilities are used by technical support personnel rather than the managers themselves. Managers must make their needs known to technically trained intermediaries who translate these needs into specifications for computer runs, make the runs, and assist in interpreting the results. This can lead to delays and readily lends itself to possible misinterpretation by support personnel of the true intentions of the managers.

A simple-to-use, interactive system would permit a manager to converse on a one-to-one basis with the model and provide an optimization process allowing for virtually immediate response to complicated network flow problems. The model developed here provides such capability for single commodity capacitated network flow problems, and can be expanded to handle more complex problems by the implementation of the enhancements listed in chapter II.

The single commodity capacitated flow model and its specializations constructed by the system are minimum cost network flow problems. The goal is to determine how (or at what rate) a good should flow through the arcs of a network to minimize shipment costs. The network is a directed graph defined by a set of m nodes, N, and a set of n arcs, A, with ordered pairs of nodes (tail,head) as elements indexed by k. For each arc there is a shipping cost per unit flow (or capacity), c_k. Each node is either a supply node where units of goods enter the network, a demand node where units leave, or transhipment node. The problem is to minimize total cost with flows, x_k, that satisfy the associated lower bounds and capacities and preserve the conservation of flow at each node:

$$\min \sum_{k \in A} c_k x_k$$

$$\text{s.t.} \sum_{k \in A \text{ with tail } i} x_k - \sum_{k \in A \text{ with head } i} x_k = b_i; \quad i \in N,$$

$$l_k \leq x_k \leq u_k; \quad k \in A$$

where b_i = (supply if i is a supply node; -demand if i is a demand node; 0 otherwise) [Ref. 3].

This interactive network model-building system was designed for the user who has only a minimal knowledge of networks. If the user can identify the fact that the problem to be solved can be represented as a single commodity capacitated flow network and has a basic knowledge of computer terminal use, then with the aid of the User's Manual (Chapter III) he should be able to effectively use the system. The system leads the user through the construction of a network by a series of simple questions. The model may then be immediately (or repeatedly) solved with GNET, a machine-independent FORTRAN program for solution of capacitated network flow problems [Ref. 3].

The features of the system may be summarized as follows:

1. The system is completly general for the class of problems at hand, allowing for a large number of diverse applications that include transport of goods, design of communications and pipeline systems, assignment of people to jobs, routing of vehicles, bid evaluation, and production planning. For a comprehensive look at state-of-the-art network model applications see [Ref. 4].

2. The user may customize the system to the specific problem being solved by the interactive input of user

defined labels for each node, naming the commodity passing through the network, and specifying the units of flow and units of cost per unit of flow.

3. The system will create an entire network or allow for the modification of single elements of a previously constructed network.

4. The data-base requires minimal storage space.

5. A series of checks are built into the system to prevent the construction of a mathematically inconsistent network.

6. The user can execute commands by a single keystroke on a terminal.

7. GNET one of the fastest and most efficient programs currently available for solving capacitated transhipment problems, thus allowing for virtually immediate solution of most network problems constructed.

8. The system is reasonably portable between different computer installations.

9. The system is designed for ease of expansion and modification.

II. SOFTWARE DESCRIPTION

A. GENERAL

The interactive network generating system consists of a driver program which co-ordinates the overall functioning of the system and eight subsystems which carry out the creation or modification of the network.

1. Data Base

The data base is structured in the same form as the data base for GNET [Ref. 3]. This structure minimizes storage but still allows for ease of manipulation. The arcs are sorted so that all arcs with the same head are stored in contiguous space. This permits the list of head nodes to be replaced by a node-length array whose jth element is the location of the first arc with head j. Thus, the network is stored as three arc-length arrays: the tail nodes T( ), the costs C( ), the minimum capacities MNCP( ), and the maximum capacities MXCP( ); also, three node-length arrays are used, the head node entries H( ) into T( ), supplies S( ) (-S( ) denotes demand), and node labels L( ). Figures 1 and 2 [Ref. 3] show the data arrays for the network described in Figures 6, 7, and 8.

Node i	[1] Head H	[2] Supply S	Label L	SCUNDEX
1	1	34	Los Angeles	L200
2	1	56	New York	N000
3	1	5	Chicago	C220
4	2	0	Omaha	O500
5	3	-5	Salt Lake City	S430
6	4	-9	Atlanta	A345
7	5	-18	Seattle	S340
8	6	-15	Denver	D516
9	9	-8	Austin	A235
10	13	-3	Minneapolis	M514
11	15	-21	Washington	W252
12	16	-16	Miami	M500
13	17			

[1] First arc i located at k = H(i).

[2] Negative supply denotes demand.

Figure 1: Node Data Arrays

Arc k	Tail T	Cost C	Minimum Capacity MNCP	[1]Maximum Capacity MXCP
1	2	34	0	11
2	3	23	0	6
3	1	28	0	10
4	2	45	5	25
5	1	57	0	21
6	5	24	0	5
7	1	56	0	7
8	4	19	0	9
9	1	61	0	5
10	2	99	0	12
11	6	48	0	3
12	3	53	0	24
13	3	26	0	8
14	4	20	0	2
15	2	14	10	23
16	6	34	0	16

[1]"*********" denotes unlimited capacity

Figure 2: Arc Data Arrays

2. Host Language

The system has been developed in FORTRAN IV Extended [Ref. 5]. FORTRAN was chosen for the following reasons:

a. FORTRAN is a general language available at most computer installations, so the system can be used with any contemporary hardware. Non-IBM systems may require some program modifications.

b. Since FORTRAN is a high-level language, the development time has been low.

c. The response time for each question is acceptable and there is little need for faster responses or enhanced efficiency.

d. Extensions and changes for the system can be easily implemented.

B. THE PROGRAM

1. Driver

The driver program initializes the system and coordinates the interaction of all the subsystems [Fig. 3]. If a new file is being created, all counters are initialized at zero and all arrays are blank, allowing for the construction of a new network. If an existing file is to be modified or completed, the old file is read into the arrays and the counters are set at the appropriate values.

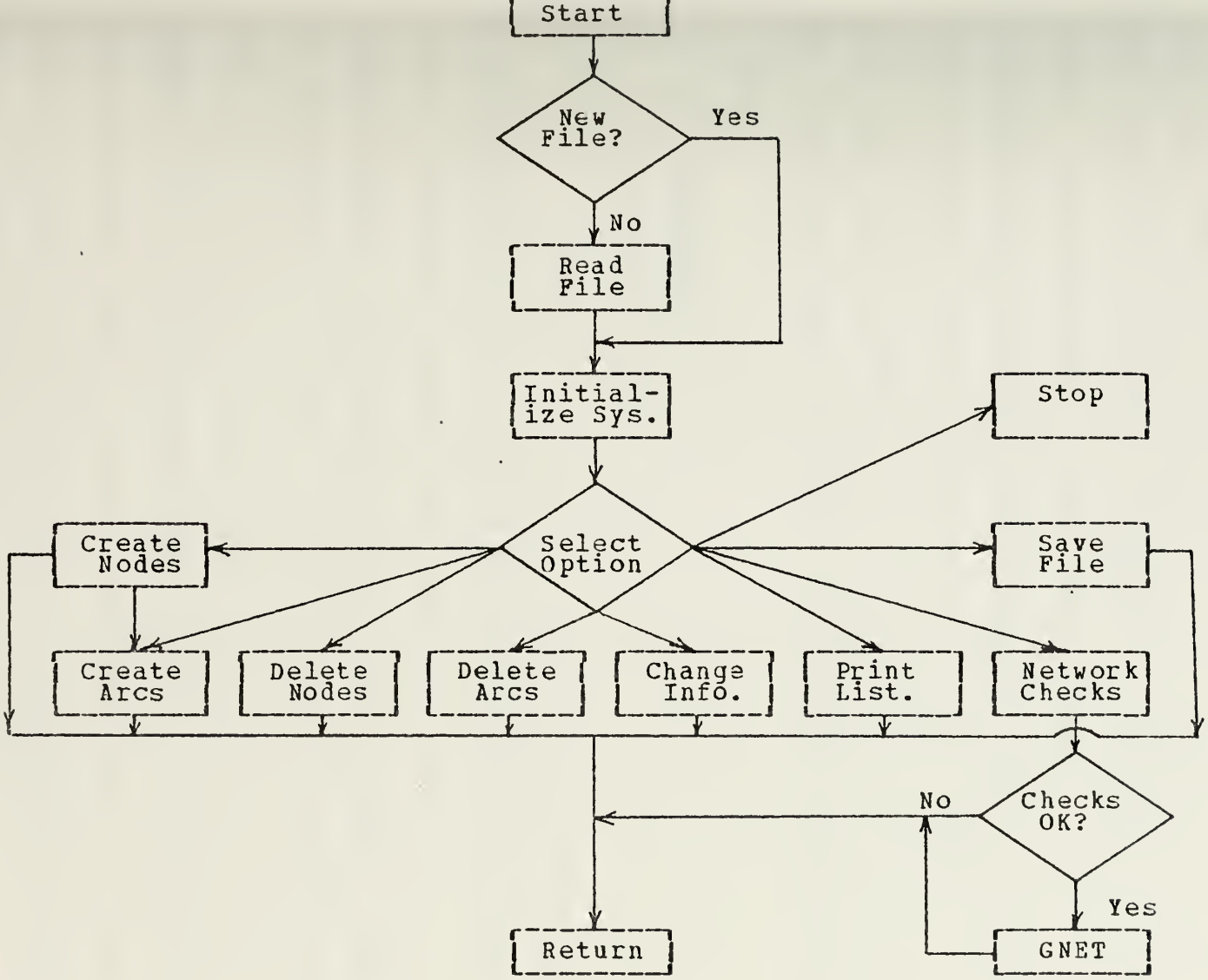

Figure 3: System Flow Chart

The overall co-ordination of the subsystems is maintained through the selection of the appropriate option from a menu for creating or updating the data file via user input. Once the action is completed, control is returned to the driver program and the user is prompted for the selection of another option. The system has been designed to allow the user to return to the option list at any time during a terminal session by using a predefined character (ie. a "?").

2. Subsystems

a. Adding Nodes

Adding a node i to the network is accomplished by first reading in the label to be assigned to the node. Prior to entering the label into the label array L( ), it is checked against all existing labels to ensure that it has not previously been used. If the check is satisfactory then the label is stored in L(i) and the head node entry array element H(i) is created where H(i) = H(i+1) indicating that the predecessor arcs, as of now, are not created. A supply is then stored in S(i), and finally the predecessor arcs are created by calling the arc-creating subroutine.

The user is also provided the capability of creating a node i "like" a node q. All data currently associated with node q is copied for node i. The user can then modify the costs, add or delete arcs, etc. for node i by returning to the option list and selecting the appropriate options to effect the changes.

b. Creating Arcs

An arc i,j can be created either in conjunction with the creation of a node or independently. If it is created in conjunction with a node, the head node index i is passed to the subroutine as the head node for the subsequent arc. If the arc is being created independently, the head node label must be entered at the terminal and its index determined by a search of the label array.

When a head or tail node label is entered during the arc creation routine, the label is checked against all other existing labels. If a match is found, the index for the node label is used in all subsequent actions concerning that node. If a match is not found, then a node is created and flagged by placing a dummy value as the supply for the node (ie. $S(i) = 2x10^9$) to indicate that the node must be formally defined (as a source, sink, or pure transhipment

node by entering a supply or demand) subsequently using the node creation option prior to completion of the network description.

Once the head and tail node indices, i and j respectively, have been determined, the arc i,j is added to the network. The arc is linked to the head node i by setting $H(q) = H(q)+1$ for all $q \geq i$. The tail node j is linked to the arc by expanding the tail array T( ) and inserting the index j at the appropriate location, $T(H(i)) = j$. This procedure maintains the contiguous relationship of all arcs with same head node and allows for an orderly expansion of the network.

c. Deleting Nodes

When deleting a node i all arcs incident to that node must be deleted as well. For this reason all incident arcs are identified, their location stored, and then listed at the terminal prior to the node being deleted. This allows the user to preview the total effect of the node deletion.

The predecessor arc indices are located directly from the head array (H(i) to H(i+1)-1).

Successor arc indices are located by a search of the tail array for the set of all k's such that $T(k) = i$. The head node index for one of the arcs k is identified by a sequential search of the head array for a node index h such that $H(h) \leq k \leq H(h+1)-1$.

After all incident arcs have been listed at the terminal, the user is provided an opportunity to abort the operation. If it is decided to abort, no change to the data base occurs. If it is decided to continue, the arcs and nodes are deleted. The arrays T( ), C( ), MNCP( ), and MXCP( ) are condensed to reflect the deletion of the arcs and similarly H( ), S( ), and L( ) are condensed to reflect the deletion of the node. The node indices in the array T( ) which are greater than the index of the node deleted must be decremented by one. The arc indices in the head array must be decremented sequentially by the number of predecessor arcs to that node that were deleted plus the number of predecessor arcs that were deleted from all head nodes listed prior to the node being decremented. Through these index manipulation procedures the network structure can be efficiently maintained in situ.

d. Deleting Arcs

When deleting an arc i,j, the arc is identified by the input of its head and tail node labels. When multiple arcs i,j exist the user must select the correct arc from the display of the respective arc costs and capacities. Once the arc is uniquely determined, it is deleted from the data base using the same procedure as that used for the deletion of an arc in conjunction with the deletion of a node.

e. Change Arc/Node Information

The spelling of a node label, the supply at a node, cost for an arc, or an arc's minimum or maximum capacity can be changed without any index manipulation other than that which is required for node or arc identification. Once the arc or node is identified by the appropriate label inputs, a listing of the data to be changed is presented at the terminal. The new data is then inserted into the appropriate arrays.

f. Printing Current Arc/Node Information

A listing of all current arc and node information can be obtained by the selection of the appropriate option. The arc list is obtained by first sequentially

selecting a node index i. All predecessor arc indices are then readily identified as the set H(i), ..., H(i+1)-1. Thus the tail node indices corresponding to the arcs are the set T(H(i)), ..., T(H(i+1)-1). The arcs are then printed at the terminal listing the tail and head node labels, cost, and minimum and maximum capacities corresponding to the appropriate index [ie. L(T(H(i+n))) TO L(i), C(H(i+n)), MNCP(H(i+n)), MXCP(H(i+n)) for n = 0, ..., H(i+1)-H(i)-1].

The node information is printed at the terminal by sequentially listing L(i) and S(i) for i = 1, ..., m.

g. Saving Data File

If the user desires a permanent file of the data arrays, they are written to a predefined data file using unformatted WRITE statements which allow for minimum storage. By listing the array lengths first, the data can be easily retrieved using READ statements that sequence the arrays in the same order as the WRITE statements. The new file will write over any previously saved file with the same file name, making permanent any changes made to the old file, or creating an entirely new file. Appendix A is a FORTRAN listing of the write statements used in the system to permit users to recover networks with other programs.

h. Network Checks and Linking GNET

Two network checks are run prior to the execution of GNET. First the supply array S( ) is scanned for flags indicating undefined nodes. When a flag is encountered the corresponding node label is printed at the terminal. Once the entire array has been checked the program returns to the option list to allow the user to formally define the listed node. If no undefined nodes are located the program continues on to the next check.

The second check is for orphan nodes, isolated nodes with no incident arcs. The head array H( ) is first checked for all nodes that have no predecessor arcs (ie. H(i) = H(i+1)). The set of node indices that have this property are then individually checked against the entire tail array T( ) to determine if there are any successor arcs to these nodes; if a node in this set has no successor arcs, its label is listed at the terminal. GNET will execute with orphan nodes in the data arrays but the user is given the option of deleting the nodes to "clean-up" the arrays or allowing them to remain as reference points for future network modifications.

After both checks are completed the system calls GNET as a subroutine to solve the final capacitated network flow problem.

3. Data Input

All user inputs are read as characters using an 80A1 format. All leading blanks are ignored. If the first character is a "?", then the program automatically returns to the option listing. This allows the user to terminate any operation at any time and return to the option list for assistance.

A null response (ie. typing "enter" without typing a value) to a request for an integer input results in a zero being assigned for the value.

For "YES/NO" responses only the first letter is read. Any first character response (including a null response) other than a "Y" is assumed to be a "NO" answer.

For each character string a SOUNDEX code is determined. This is an alpha-numeric code used as a spelling error check [eg., Ref. 6]. With reasonable accuracy a misspelled word's SOUNDEX code will match the code of the correct spelling of the word. When a search of the label array L( ) is conducted the SOUNDEX codes are checked as

well. If the spelling of the newly typed label does not exist in the array but there is a SOUNDEX code match then a possible spelling error is assumed. The user is given the options of indicating no spelling error exists at which time the label is inserted into the array, or correcting the label currently in the data base with the newly typed label, or vice versa.

After the SOUNDEX code has been determined for a label, the character string is stored in a two dimensional node-length array. Up to twenty characters are saved for each label entry. The last four characters are used for the SOUNDEX code.

C. FUTURE ENHANCEMENTS

Due to the complexity involved in expanding network flow problems to take into consideration more general "real world" applications there are no "easy to use" general interactive network generating systems in wide use today. For this reason the expansion or enhancement of this system is open to a myriad of possibilities. For instance:

1. Expand the data base to create multicommodity network flow problems [Ref. 1].

2. Customize the system for multiperiod problems with intermediate inventory at the nodes.

3. Customize the system so that production levels can be optimized at manufacturing nodes as well as flows to demand nodes.

4. Allow for mixed integer programming problems where there are both fixed and variable costs on the arcs.

III. USER'S MANUAL

A. INTRODUCTION

The interactive network generating system is designed to create, update, and solve a single-commodity network with only a minimal knowledge of network structure and only a rudimentary knowledge of computer terminal operation. This is accomplished through the interactive use of a set of FORTRAN programs which lead the user, step-by-step, through the construction of the network by a series of questions and then links with GNET, a machine independent FORTRAN program for the solution of capacitated network flow problems [Ref. 3].

The entire system has been developed in FORTRAN language to allow for portability and possible use with other network optimization packages.

The questions asked by the system are basically in English sentence structure and self-explanatory in nature.

A system of checks have been developed for the program that may prevent the user from creating a mathematically inconsistent network.

Appendix B is a dictionary of network terminology to assist the user in understanding the questions asked while creating the network.

B. INPUT FORMATS

All numerical inputs into the system must be integer values. They are entered one value per line in "free format" (a value can be entered anywhere on a line); "right-justification" is not necessary. If an alpha character is entered where a numeric character is expected, the following warning is given:

YOU HAVE ENTERED A NONINTEGER WHERE AN INTEGER IS REQUIRED. REINPUT THE CORRECT VALUE.

A null response for a request for an integer input is assumed to be a zero value.

A label input will truncate beyond twenty characters. All labels must be unique (no two nodes may have the same label) or a warning is given as is indicated in the following example:

NEW YORK HAS ALREADY BEEN DEFINED AS A NODE NAME. EITHER INPUT ANOTHER NAME OR TYPE ? FOR ASSISTANCE.

Only a "Y" need be entered for a yes answer to any question requiring a "YES/NO" response. Any other response (including a null response) will be considered as a "NO" answer.

C. INITIALIZING THE SYSTEM

Upon starting the system the user is first presented a listing of system requirements followed by the question:

ARE YOU CREATING A NEW FILE? YES/NO

A "YES" answer initializes the system and starts the construction of the network "from scratch". In this case, the user is then asked,

WHAT IS THE NAME OF THE COMMODITY PASSING THROUGH THE NETWORK?
>STEEL ("<" indicates a keyboard prompt for user.)
IN WHAT UNITS ARE THE FLOWS AND CAPACITIES MEASURED (IE. LBS, TONS, ETC)?
>TONS
IN WHAT UNITS ARE THE COSTS GIVEN (IE. $/TONS)?
>$100/TONS

The name and unit labels are used in the future questions to aid in customizing the system for the individual user. The user must keep in mind that once a unit type has been defined, all future capacities, flows, and costs must be expressed accordingly.

A "NO" answer to the new file question results in the reading in of previously constructed network data arrays from a file that was created and saved during an earlier terminal session. This allows the user to create only part of the network at a time or change existing network data arrays.

The dimensioning of the data arrays used in the system is dynamic, thus the user is asked,

```
THE DIMENSIONS OF THE SYSTEM ARE FOR 1000 ARCS AND 500
NODES.  DO YOU WISH TO CHANGE THEM? YES/NO
>YES
HOW MANY NODES DO YOU WISH TO CREATE?
>20
HOW MANY ARCS DO YOU WISH TO CREATE?
>60
DATA REGION REQUIRED = 580    AVAILABLE = 12000
```

The dimensions once set will prevent the user from increasing the size of the network beyond the size of the user-defined values. Therefore the values should be larger than the actual dimensions of the network to be constructed to allow for errors or future expansion. When creating a new network default dimensioning is 500 nodes and 1000 arcs.

D. MENU FOR CREATING/UPDATING DATA FILE

Once the system is initialized either as a new network or with the parameters of a previously constructed network, the user is presented a menu of choices from which to create or update the network [Fig. 4]. Typing the corresponding line number, the program will present the appropriate series of questions to allow the user to accomplish the desired action.

Typing a "?" at any time will return the user to this menu, thus allowing for the selection of another option to aid in the resolution of a possible problem.

1. Create Nodes

Creating nodes is the usual starting point for the construction of a new network. The questions asked are basically self-explanatory. The following is an example of creating the twelfth node in a network with the commodity being STEEL [Fig. 7]:

1. CREATE NODES
2. CREATE ARCS
3. DELETE EXISTING NODES
4. DELETE EXISTING ARCS
5. CHANGE CURRENT NODE/ARC INFORMATION WITHOUT CHANGING FILE INFORMATION
6. PRINT LISTING OF CURRENT FILE INFORMATION
7. SAVE A FILE
8. STOP

Figure 4: Menu for Creating/Updating Data File

HOW MANY NODES DO YOU WANT TO CREATE?

>1

WHAT IS THE NAME OF NODE NO. 12?

>MIAMI

DO YOU WANT MIAMI TO BE CREATED LIKE A PREVIOUSLY CREATED NODE (IE. ARCS TO AND FROM THE SAME LOCATIONS)? YES/NO

>NO

WHAT IS THE SUPPLY OF STEEL AT MIAMI IN TONS?

>0

WHAT IS THE DEMAND FOR STEEL AT MIAMI IN TONS?

>16

HOW MANY ARCS SHIPPING TO MIAMI DO YOU WANT TO CREATE?

>1

WHAT IS THE NAME OF THE TAIL NODE OF ARC NO. 1 THAT SHIPS TO MIAMI?

>ATLANTA

COST PER UNIT IN $100/TON?

>34

UNLIMITED CAPACITY? YES/NO

>NO

MAX. CAPACITY IN TONS?

>16

MIN. CAPACITY IN TONS?

>0

The questions concerning supply and demand are related in such a way that if supply is greater than zero the user is not allowed to enter a demand. If supply is zero then a demand may be entered. If demand is also zero it is assumed that the node is a pure transhippment node.

A negative supply is interpreted as a demand. A negative demand is interpreted as a supply.

Only arcs shipping to the node being created can be constructed at this time. If the tail node has not yet been constructed the program will generate a dummy node with the same label as entered for the tail node. This dummy node must be formally created using the "ADD NODES" option prior to completion of the network description.

All arcs leading away from the node being created are constructed in conjunction with their head node at a later time.

The user is also provided the option of creating a node "like" a previously created node (ie. arcs to and from the same locations, same costs, and same capacities) thus allowing for a minimal amount of work for the construction

of two or more similar nodes. By answering "YES" to the previous question the following action is taken:

WHAT IS THE NAME OF THE NODE YOU WANT MIAMI TO BE LIKE?

>AUSTIN

MIAMI NOW HAS THE FOLLOWING STRUCTURE:

DEMAND: 3 TONS

ARCS: TO	FROM	COST $100/TON	MIN.CAP. TONS	MAX.CAP. TONS
MIAMI	LOS ANGELES	61	0	5
MIAMI	NEW YORK	99	0	12
MIAMI	ATLANTA	49	0	3
MIAMI	CHICAGO	53	0	24

TO MODIFY ANY OF THE ABOVE DATA RETURN TO THE MENU AND SELECT THE APPROPRIATE OPTION.

2. Create Arcs

Arcs can be constructed independent of the creation of a node. The same questions and restrictions are required as for the construction of an arc in conjunction with the creation of a node (see previous example).

3. Delete Existing Nodes

Deleting a node can cause the greatest change in a network's structure, thus it is also the most probable

operation to cause an error. All arcs incident to the node must be deleted as well. This can isolate other nodes leaving them completely disjoint from the rest of the network. For this reason the program will list the arcs omitted by the deletion of a node and allow the user the option to abort the operation prior to actual execution.

The following is an example of deleting a node:

```
WHAT IS THE NAME OF THE NODE YOU WANT TO DELETE?
>ATLANTA
DELETING ATLANTA RESULTS IN THE DELETION OF THE FOLLOWING
ARCS:
      NEW YORK      TO  ATLANTA
      ATLANTA       TO  AUSTIN
      ATLANTA       TO  MIAMI
DO YOU STILL WANT TO DELETE THIS NODE? YES/NO
```

A "YES" response will cause the deletion of the node as well as the deletion of the listed arcs from the data arrays. A "NO" response will cause no change to the network to occur.

4. Delete Existing Arcs

Since all arcs can be identified by their head and tail nodes only these labels are required to be entered in

order to delete a given arc as is indicated by the following example:

```
WHAT IS THE NAME OF THE HEAD NODE OF THE ARC YOU ARE
DELETING?
>CHICAGO
WHAT IS THE NAME OF THE TAIL NODE OF THE ARC YOU ARE
DELETING?
>NEW YORK
```

5. Change Current Arc/Node Information

This option allows the user to change, update, or correct any element within the data base as long as the node-arc structure of the network is not altered. Upon selecting this option, the user is presented the menu in Figure 5 from which to select the type of change desired. After the completion of a change, the program returns to the "MENU OF CHANGES". To return to the "MENU FOR CREATING/UPDATING DATA FILE" enter a "?" at any time.

a. Change/Correct Spelling of Node Label

To change the spelling of a node label (ie. a misspelled label) the user is simply asked what label is to be changed and the new spelling as indicated in the following example:

1. CHANGE/CORRECT SPELLING OF NODE LABEL

2. CHANGE THE AMOUNT OF SUPPLY OR DEMAND AT A NODE

3. CHANGE COST FOR AN ARC

4. CHANGE CAPACITY FOR AN ARC

Figure 5: Menu of Changes

WHICH NODE LABEL DO YOU WANT TO CHANGE?

>DENVAR

WHAT IS THE NEW LABEL?

>DENVER

b. Change Amount of Supply or Demand

When changing the amount of supply or demand at a node the user must first identify the node. This results in the display of a listing of the current supply or demand. The user then enters the new value. The following is an example of changing a demand node to a pure transhippment node:

WHAT IS THE NAME OF THE NODE FOR WHICH YOU WANT TO CHANGE SUPPLY OR DEMAND?

>AUSTIN

THE DEMAND IS CURRENTLY 8 TONS.

WHAT IS THE NEW SUPPLY AT THIS NODE?

>0

WHAT IS THE NEW DEMAND AT THIS NODE?

>0

c. Change Cost for an Arc

When changing the cost for an arc the user must first identify the arc to be changed. This results in the display of a listing of the current cost. The user then enters the new value. The following is an example of changing the cost on an arc:

```
WHAT IS THE NAME OF THE HEAD NODE OF THE ARC ON WHICH YOU
WANT TO CHANGE COST OR CAPACITY?
>AUSTIN
WHAT IS THE NAME OF THE TAIL NODE OF THIS ARC?
>LOS ANGELES
CURRENTLY, COST:          61 $100/TON
           MAX. CAPACITY:   5 TONS
           MIN. CAPACITY:   0 TONS
WHAT IS THE NEW COST?
>75
```

d. Change Capacity of an Arc

Changing the capacity of an arc is done in the same manner as is done for changing the cost for an arc. Only the final question changes as follows:

```
IS THE NEW MAX. CAPACITY UNLIMITED? YES/NO
>NO
```

```
WHAT IS THE NEW MAX. CAPACITY?
>7
WHAT IS THE NEW MIN. CAPACITY?
>0
```

6. Print Listing of Current File Information

This option allows the user to print a listing of node and arc information currently in the data base. The user is given the option of printing the listing at the terminal and/or printer. Figures 6 and 7 are examples of the listings for a given network. Figure 8 is a graphical representation of the network.

7. Save a File

Either at the end of a terminal session or periodically during the session the user can save the file just completed by selecting option seven. This causes the program to write over any previously saved file thus creating a new file or correcting all items that have been changed prior to the save option execution. This allows the user to create a permanent file that can be read by the system at a later date. Only through the execution of this option will any change to the permanent file occur.

ARC	FROM	TO	COST $100/TON	MIN CAP TONS	MAX CAP TONS
1	NEW YORK	CHICAGO	34	0	11
2	CHICAGO	OMAHA	23	0	6
3	LOS ANGELES	SALT LAKE CITY	28	0	10
4	NEW YORK	ATLANTA	45	5	25
5	LOS ANGELES	SEATTLE	57	0	21
6	SALT LAKE CITY	DENVER	24	0	5
7	LOS ANGELES	DENVER	56	0	7
8	OMAHA	DENVER	19	0	9
9	LOS ANGELES	AUSTIN	61	0	5
10	NEW YORK	AUSTIN	99	0	12
11	ATLANTA	AUSTIN	48	0	3
12	CHICAGO	AUSTIN	53	0	24
13	CHICAGO	MINNEAPOLIS	26	0	8
14	OMAHA	MINNEAPOLIS	20	0	2
15	NEW YORK	WASHINGTON	14	10	23
16	ATLANTA	MIAMI	34	0	16

Figure 6: Arc Report

NODE	NAME	SUPPLY TONS
1	LOS ANGELES	34
2	NEW YORK	56
3	CHICAGO	5
4	OMAHA	0
5	SALT LAKE CITY	-5
6	ATLANTA	-9
7	SEATTLE	-18
8	DENVER	-15
9	AUSTIN	-8
10	MINNEAPOLIS	-3
11	WASHINGTON	-21
12	MIAMI	-16

A NEGATIVE SUPPLY IMPLIES DEMAND

Figure 7: Node Report

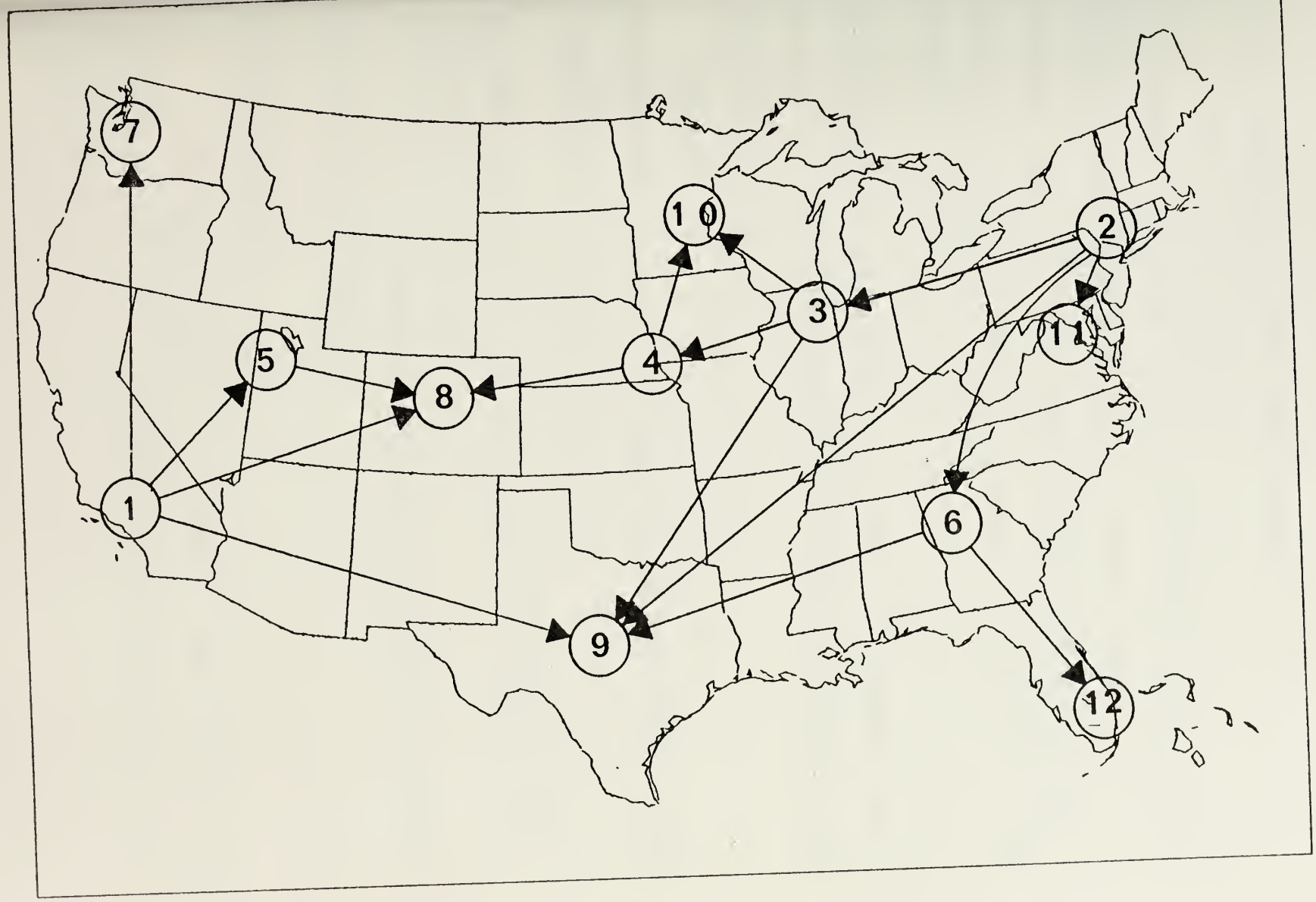

Figure 8: Graphical Representation of Example Network

8. Run Network Checks and Link GNET

Prior to the execution of GNET two checks are run on the network. The first check is to determine whether all nodes have been defined using the "CREATE NODES" option. All dummy nodes that were created while constructing arcs and not subsequently defined are listed as indicated in the following example:

THE FOLLOWING NODES HAVE NOT EXPLICITLY BEEN DEFINED (IE. IT HAS NOT BEEN DETERMINED WHETHER THE NODE IS A SOURCE, SINK, OR PURE TRANSHIPMENT):

MONTEREY

YOU MUST ADD THE NODES EXPLICITLY TO THE DATA BASE TO ALLOW A SOLUTION TO THE NETWORK TO BE FOUND.

The user must create these nodes before the program will allow execution of GNET.

If all nodes are defined the user will be informed by the following message:

ALL NODES DEFINED

The second check is for orphan nodes, isolated nodes with no incident arcs. The user is presented with the following listing:

THE FOLLOWING NODE(S) ARE ORPHANS (IE. NO INCIDENT ARCS):

MIAMI

GNET WILL WORK WITH THESE NODES LEFT IN THE DATA BASE, BUT DO YOU WANT THEM DELETED? YES/NO

A "YES" response will cause the listed nodes to be permanently deleted from the data base. The deletion will not effect the solution to the network problem. A "NO" response will leave the nodes in the data arrays, thus allowing the user to add arcs to these nodes at a later date without formally recreating the nodes.

Once the checks are successfully completed the program sends the completed network data base to GNET. GNET performs some additional checks, such as for improper scaling of cost coefficients, problem feasibility, etc. For more information concerning the internal operation of GNET see [Ref. 3]. If all network checks are successful the final optimal solution will be listed as in Figures 9 and 10.

9. Stop

The "STOP" option simply terminates the terminal session, but prior to the physical termination the user is provided one more opportunity to "SAVE A FILE".

FROM	TO	COST $100/TON	MIN.CAP TONS	FLOW TONS	MAX.CAP TONS
NEW YORK	CHICAGO	34	0	10	11
CHICAGO	OMAHA	23	0	6	6
LOS ANGELES	SALT LAKE CITY	28	0	10	10
NEW YORK	ATLANTA	45	5	25	25
LOS ANGELES	SEATTLE	57	0	18	21
SALT LAKE CITY	DENVER	24	0	5	5
LOS ANGELES	DENVER	56	0	4	7
OMAHA	DENVER	19	0	6	9
LOS ANGELES	AUSTIN	61	0	2	5
NEW YORK	AUSTIN	99	0	0	12
ATLANTA	AUSTIN	48	0	0	3
CHICAGO	AUSTIN	53	0	6	24
CHICAGO	MINNEAPOLIS	26	0	3	8
OMAHA	MINNEAPOLIS	20	0	0	2
NEW YORK	WASHINGTON	14	10	21	23
ATLANTA	MIAMI	34	0	16	16

COST= 4723.

Figure 9: Final Solution Arc Report

NODE	NODE LABEL	SUPPLY TONS	DUAL VARIABLES $100/TON
1	LOS ANGELES	34	-26
2	NEW YORK	56	0
3	CHICAGO	5	-34
4	OMAHA	0	-63
5	SALT LAKE CITY	-5	-54
6	ATLANTA	-9	-45
7	SEATTLE	-18	-83
8	DENVER	-15	-82
9	AUSTIN	-8	-87
10	MINNEAPOLIS	-3	-60
11	WASHINGTON	-21	-14
12	MIAMI	-16	-79

Figure 10: Final Solution Node Report

E. ADDITIONAL SYSTEM CHECKS AND SAFEGUARDS

1. Spelling Errors

A new label is always checked with existing labels already in the data base prior to its insertion in the data base. A spelling error check is made by the comparison of a coded form of the labels [Ref. 6]. If the codes match but the exact spelling of the labels do not, a possible spelling error is assumed and the user is given the option to correct the error. This helps prevent the creation of two nodes, due to a spelling error, when the user intends only one. The following is presented to the user to correct the probable error:

YOU HAVE A POSSIBLE SPELLING ERROR. TYPE THE CORRESPONDING LINE NUMBER FOR THE APPROPRIATE ACTION.

1. NO SPELLING ERROR EXISTS
2. MISSPELLED: CHICEGO CORRECT WITH: CHICAGO
3. MISSPELLED: CHICAGO CORRECT WITH: CHICEGO

A response of "1" will cause no change to occur and the program will continue. A response of "2" or "3" will cause the first label to be replaced by the second permanently.

2. Minimum Greater Than Maximum

If the minimum capacity is greater than the maximum for a given arc the following warning is displayed:

YOU HAVE A MIN. CAPACITY GREATER THAN THE CORRESPONDING MAX. PROVIDE THE CORRECT VALUES.

Simply provide the correct values in the proper order when prompted for the new maximum and minimum capacities.

3. Multiple Arcs

Both the network generating system amd GNET will admit multiple arcs. When the system is required to identify a specific multiple arc the user must identify the correct arc by its corresponding cost or capacity as is indicated in the following example:

THERE ARE MULTIPLE ARCS BETWEEN THE NODES NEW YORK AND CHICAGO. TYPE THE CORRESPONDING LINE NUMBER OF THE ARC YOU WANT DELETED.

1. COST: 34 $100/TON, MAX.CAP.: 11 TONS, MIN.CAP.: 0 TONS
2. COST: 25 $100/TON, MAX.CAP.: 7 TONS, MIN.CAP.: 0 TONS

APPENDIX A

```
      IMPLICIT INTEGER(A-Z)
      ...

      REWIND 09
      WRITE(9)  N, M, IDIM1, IDIM2, (COMOD(I),I=1,5),
     1(CAPLAB(J1),J1=1,2), (CSTLAB(J2),J2=1,2)
      WRITE(9)  (T(J3),J3=1,N), (C(J4),J4=1,N),
     1(MXCP(J5),J5=1,N), (MNCP(J6),J6=1,N), (CP(J7),J7=1,N)
      WRITE(9)  (H(J8),J8=1,M), (S(J9),J9=1,M)
      WRITE(9)  ((L(J10,J11),J10=1,6),J11=1,M)
```

Number of arcs:	N
Number of nodes:	M
Commodity label:	COMOD
Capacity units label:	CAPLAB
Cost units label:	CSTLAB
Tail nodes:	T
Costs:	C
Maximum capacities:	MXCP
Minimum capacities:	MNCP
Maximum - Minimum capacities:	CP
Head node entries into T( ):	H
Supplies:	S
Node labels:	L

APPENDIX B

Arc A vector, a unidirectional means of indicating commodity transportation among locations, or routes.

Alpha Character Any letter in the English alphabet (ie. A, B, C, ..., Z).

Cost Per Unit The cost of shipping one unit of a commodity across a given arc.

Dual Variable A price associated with a constraint of the original (primal) problem (ie. marginal cost or shadow price).

Head Node An incident node at the end of an arc to which the arc flow is moving.

Integer Whole number.

Max. Capacity/Upper Bound The maximum amount of a commodity that can flow across a given arc.

Min. Capacity/Lower Bound The minimum amount of a commodity that can flow across a given arc.

Multiple Arcs More than one arc between the same two nodes with their flow orientation in the same direction.

Node A location or terminal connected by arcs and served by whatever physical means of transportation is associated with the arcs.

Numeric Character A single digit (ie. 0, 1, ..., 9).

Orphan Node An isolated node with no incident arcs.

Predecessor Arc For a given node an arc with its head at that node.

Prompt Indicated in this text as ">", the prompt character indicates that a user-supplied input is required before further execution.

Sink Node A node with demand.

Source Node A node with supply.

Successor Arc For a given node an arc with its tail at that node.

Transhippment Node A node with zero supply and zero demand.

LIST OF REFERENCES

1. Geoffrion, A.M. and Graves, G.W., "Multicommodity Distribution System Design By Benders Decomposition," Management Science, v. 20, p. 822-844, January 1974.

2. Geoffrion, A.M. and Powers, R.F., "Management Support Systems," The Wharton Magazine, v. 5, p. 49-59, Spring 1981.

3. Bradley, G.H., Brown, G.G., and Graves, G.W., "Design and Implementation of Large Scale Primal Transshipment Algorithms," Management Science, v. 24, p. 1-34, September 1977.

4. Bradley, G.H., "Survey of Deterministic Networks," AIIE Transactions, p. 222-234, September 1975.

5. International Business Machines Inc., IBM/360 and System/370 FORTRAN IV Language, GC-28-6515, May 1974.

6. Knuth, D., The Art of Computer Programming, v. 3: Sorting and Searching, Addison-Wesley, p. 391, 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Professor Gerald G. Brown, Code 55BW Naval Postgraduate School Monterey, California 93940	23
5.	Professor Glenn W. Graves Graduate School of Management University of California Los Angeles, California 90024	1
6.	CAPT Charles S. Burchinal, USMC 9004 Rollingwood Dr. Oxon Hill, Maryland 20022	1

Thesis 194423
B859 Burchinal
c.1 IVONNE: an inter-
active network model-
building system.